1 Quick Start

Install USB Drivers
- Run `CP2101_Drivers.exe` which is located in the `Drivers\USB` directory of the ezLCD CD.
- Connect ezLCD to your computer through the USB cable
- Proceed with Plug-and-Play installation

The ezLCD COM port assignment
- Connect ezLCD to your computer through the USB cable
- The ezLCD COM port assignment should be shown in th Device Manager as:
  CP2101 USB to UART Bridge Controller (COMx)
  where x is the COM port number

Power ON/OFF buttons of the ezLCD-002
- The ezLCD-002 power can be turned ON by pressing the ON pushbutton on the back
- The ezLCD-002 power can be turned OFF by pressing the OFF pushbutton on the back
- The ezLCD power options are describet in the chapter: Power Supply & Distribution

Touch Screen
- Make sure that the USB drivers are installed
- Connect ezLCD to your computer through the USB cable
- Verify the ezLCD COM port assignment
- Power ON the ezLCD
- Make sure that the ezLCD COM port is not opened by other applications
- Run `ezLCDtch.exe`, which is located in the `Utilities\ezLCDtch` directory of the ezLCD CD.
- Select the COM port assigned to the ezLCD
- Press the **Open** button
- The ezLCD touch screen taps should now be reflected by the `ezLCDtch` utility

Example of sending commands to the ezLCD using a Windows Command Prompt
- Make sure that the USB drivers are installed
- Connect ezLCD to your computer through the USB cable
- Verify the ezLCD COM port assignment. Let's **assume** that ezLCD is assign to the COM5
- Power ON the ezLCD
- Make sure that the ezLCD COM port is not opened by other applications
- Open Command Prompt on your computer
- Set ezLCD Com port parameters. Type:
  `MODE COM5 BAUD=115200 PARITY=N DATA=8 STOP=1`
- Turn the ezLCD Light ON. Type:
  `ECHO "" > COM5`
- Set current ezLCD color to green. Type:
  `ECHO $8 > COM5`
- Send CLS command Type:
  `ECHO ! > COM5`
  The ezLCD will fill it's entire screen with the green color
- Turn the ezLCD Light OFF. Type:
  `ECHO # > COM5`
- etc.

2 ezLCD-002

2.1 Overview

Congratulations on your purchase of ezLCD-002!

The ezLCD-002 is an all-in-one advanced color TFT LCD panel which includes:

- 240x160 pixel, 512 color, 2.7" TFT LCD (Sony ACX705AKM)
- LCD controller (Epson SED1375)
- Embedded processor (Atmel ATmega128L)
- Power supply, which generates all the voltages needed by the logic and the display itself
- Lithium-Ion battery charger
- Touch screen
- Interface drivers and other circuitry

The ezLCD-002 communicates with the outside world through several implemented interfaces:

- RS232
- USB
- I2C
- SPI
- 8 bit parallel (Centronix printer protocol)

The ezLCD-002 is driven by a set of commands, which can be fed through any of the implemented interfaces. The device may be used as an "intelligent" display, or as a stand alone device. There is plenty of flash memory left in ATmega128 to incorporate additional graphical instructions, or to customize the software for particular tasks. Possible applications include automotive, avionics, nautical, industrial control, hobby, etc.

Note: This is a preliminary documentation.

2.2 Operation

The ezLCD-002 is driven by a set of 8 bit commands, which can be received by any of the implemented interfaces.

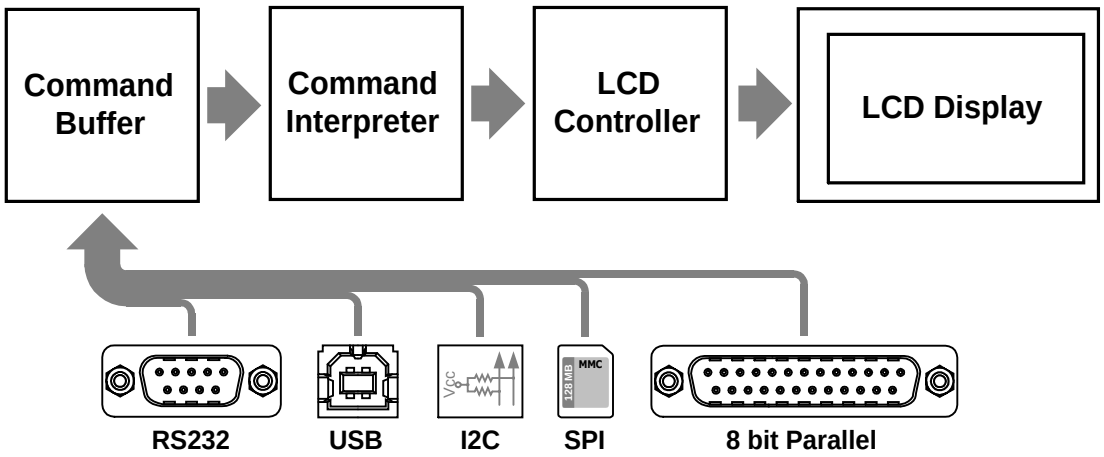


Figure 3. ezLCD-002 Data flow Diagram

Each of the implemented interfaces uses the same set of ezLCD Commands.
Upon arrival, the ezLCD Commands are stored into the 1024 byte long **Command Buffer** as shown in *Figure 3*.
All interfaces use the same Command Buffer. The **Command Interpreter** (*Figure 3*.), picks up byte-by-byte the commands stored in the Command Buffer and drives the **LCD Controller** with the corresponding set of signals and instructions. The commands are processed on a First-In, First-Out principle.
This data flow architecture makes possible the implementation of some advanced graphical commands, like CIRCLE_R, LINE_TO_XY, PUT_BITMAP, etc.

Example:

The following commands will draw a green circle with a radius of 60 pixels, and a centered position at column 120, row 80.

Pseudo-Code (ANSI C format):

```
SetColor(GREEN);   /* Set the drawing color to green */
SetXY(120, 80);    /* Set the position to x = 120, y = 80 */
CircleR(60);       /* Draw the circle with the radius of 60 pixels */
```

Data sent to the ezLCD (Columns: Value and Format)**:**

Mnemonic	Value	Format	Comment
SET_COLOR	24	hex	Set the drawing color to:
green	00111000	bin	green
SET_XY	25	hex	Set the drawing position to:
120	120	dec	x (column) = 120
80	80	dec	y (row) = 80
CIRCLE_R	29	hex	Draw the circle with the radius of:
60	60	dec	60 pixels

2.3 Hardware & Interfaces

2.3.1 Block Diagram

The ezLCD-002 Hardware Block Diagram is shown in *Figure 4.* below.

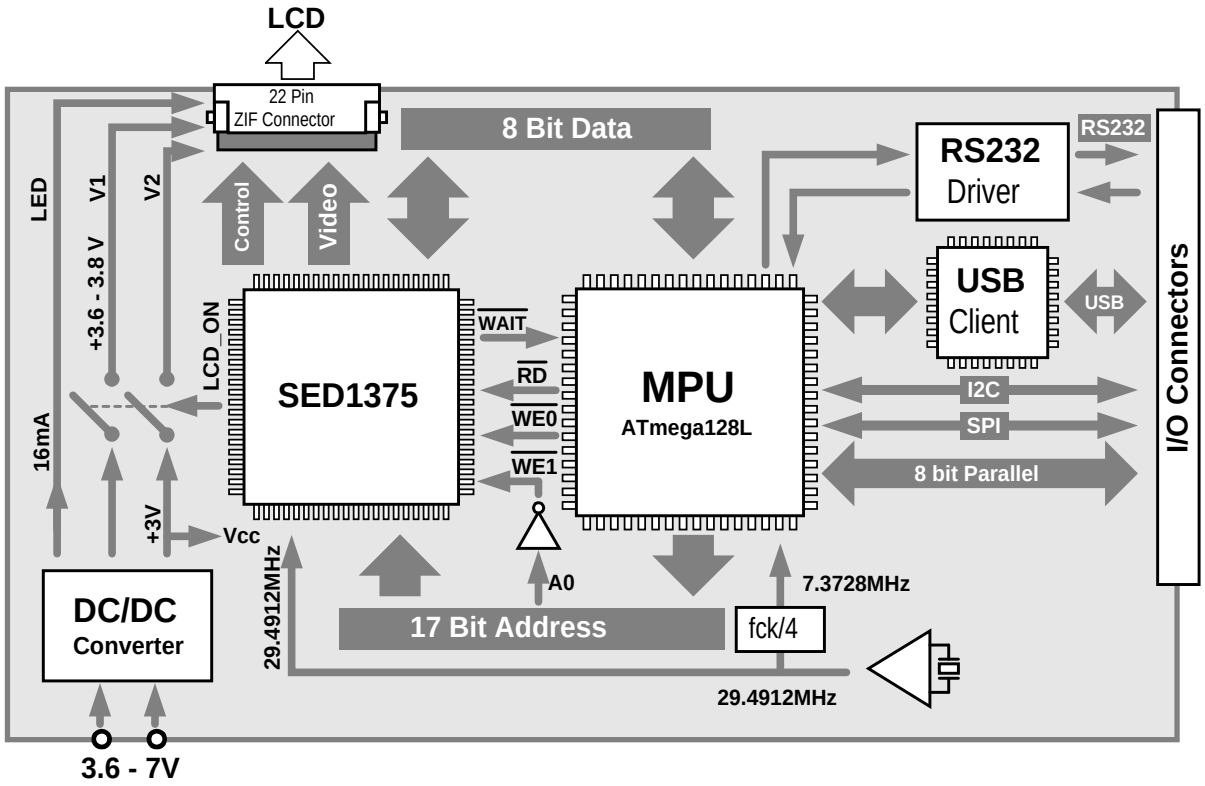


Figure 4. ezLCD-002 Block Diagram

The ezLCD-002 receives commands through any of the available interfaces (RS232, USB, I2C, SPI and Parallel).
The MPU (ATmega128L) processes the received data and writes the resulting pixels into the Video RAM of the SED1375 LCD controller.
The SED1375 generates the "Digital CRT" video signals, using the data stored in the Video RAM.

2.3.2 Pin Configuration

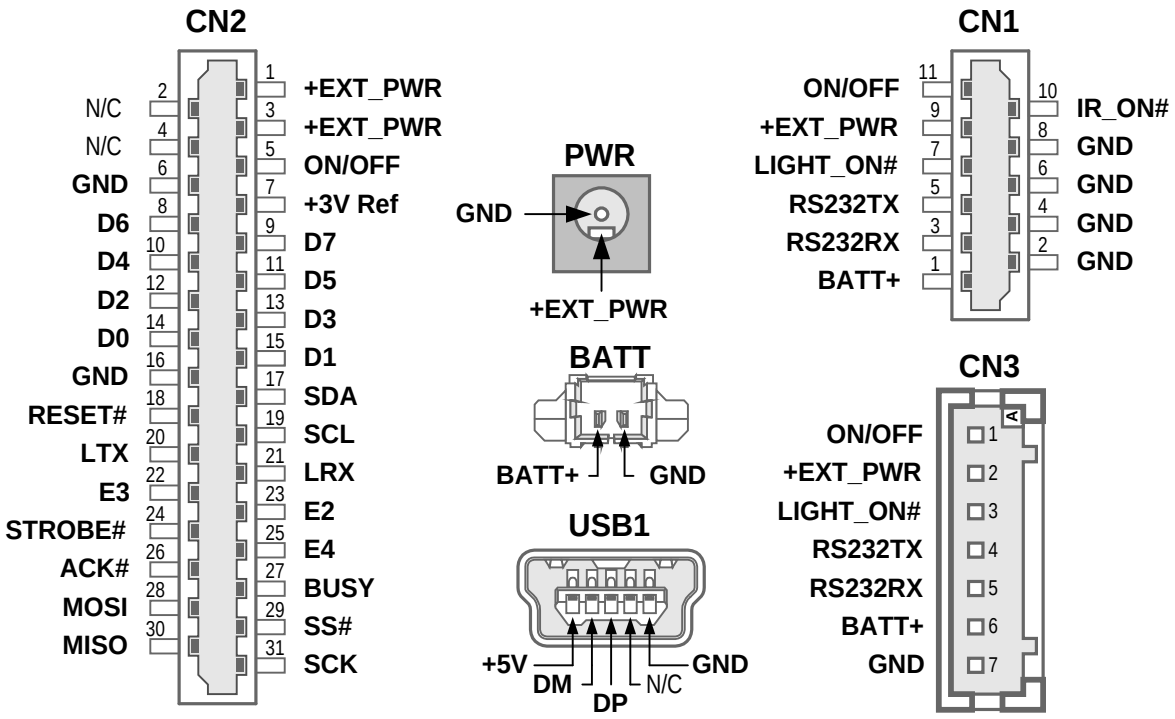


Figure 5. LCD-002 Connectors

The table below describes the pins and signals of the ezLCD-002.
Since the ezLCD-002 uses the ATmega128 microcontroller by Atmel, the table also shows the corresponding ATmega128 names for applicable pins.

Pin Name	ATmega128 Pin Names	Connector	Type	Description
+3V Ref	N/A	CN2	Output	I/O reference voltage. may be used as a pull-up source (I2C etc.). It SHOULD NOT be used as a power source.
+5V	N/A	USB1	Input Pwr.	USB VBUS Signal.
+EXT_PWR	N/A	CN1,CN2, CN3, PWR	Input Pwr.	External power voltage. Min = +3.6V Max = +7.0V
ACK#	PE7	CN2	Output	Acknowledge signal of the Parallel Interface. Active low. Min = 0V Max = +3V
BATT+	N/A	CN1, CN3, BATT	Input Pwr.	Lithium-Ion battery +3.6V
BUSY	PE6	CN2	Output	Busy signal of the Parallel Interface. Active high. Min = 0V Max = +3V

This table is continued on the next page

Pin Name	ATmega128 Pin Names	Connector	Type	Description
D0 -D7	PF0 - PF7	CN2	Input	Data inputs of the Parallel Interface. Min = 0V Max = +3V or Open
DM	N/A	USB1	I/O	USB Data Minus
DP	N/A	USB1	I/O	USB Data Plus
E2 - E4	PE2 - PE4	CN2	I/O	Spare
GND	GND	CN1, CN2, CN3, USB, PWR, BATT	Gnd	Ground
IR_ON#	N/A	CN1	Output	Drain of the IRLML2502 N-Channel MOSFET, capable of sinking up to 4A. The source of the IRLML2502 is connected to GND. This pin may be used to drive hi-power infrared transmitter. The gate of the IRLML2502 is connected to PB7 pin of the ATmega128L.
LIGHT_ON#	D7	CN1, CN3	Input	Light On signal. Active Lo. When connected to GND turns on the LCD front light. The function of this signal is identical to LIGHT_ON and LIGHT_OFF commands. Min = Gnd Max = +3V or Open
LRX	PE0	CN2	Input	RS232 TTL Input Min = 0V Max = 3V
LTX	PE1	CN2	Output	RS232 TTL Output Min = 0V Max = 3V
MISO	PB3	CN2	I/O	SPI Master Input Slave Output signal Min = 0V Max = 3V
MOSI	PB2	CN2	I/O	SPI Master Output Slave Input signal Min = 0V Max = 3V
N/C	N/A	CN2, USB1	N/A	Not Connected
ON/OFF	N/A	CN1, CN2, CN3	Input	+3.6 to +7V turns ON the ezLCD-002 power. 0 to +1V turns OFF the ezLCD-002 power. Open leaves the ezLCD-002 power unchanged. Tmin = 1ms Rin > 250 kOhm.
RESET#	RESET	CN2	Input	Hardware reset. Active low. Min = 0V Max = +3V or Open

This table is continued on the next page

Pin Name	ATmega128 Pin Names	Connector	Type	Description
RS232RX	N/A	CN1,CN3	Input	RS232 Receive Min = -12V Max = +12V
RS232TX	N/A	CN1, CN3	Output	RS232 Transmit Min = -12V Max = +12V
SCK	PB1	CN2	I/O	SPI Serial Clock. Min = 0V Max = +3V
SCL	PD0	CN2	I/O	I2C Serial Clock Min = 0V Max = +3V
SDA	PD1	CN2	I/O	I2C Serial Data Min = 0V Max = +3V
SS#	PB0	CN2	I	SPI Slave Select. This signal is used only when the ezLCD-002 is configures as SPI Slave. Active low. Min = 0V Max = +3V **Note:** During reset and power-up this signal is used as PGM#.
STROBE#	PE5	CN2	Input	Strobe signal of the Parallel Interface. Active low. Min = 0V Max = +3V or Open

2.3.3 Power Supply & Distribution

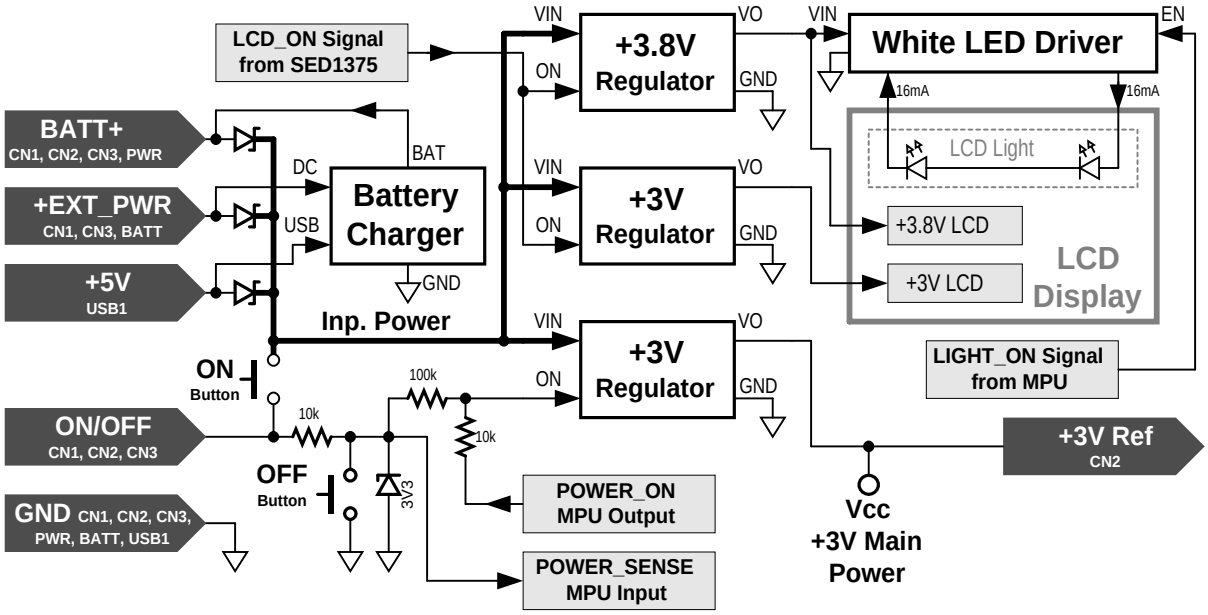


Figure 6. ezLCD-002 Power Supply and Distribution

Power Sources

The ezLCD-002 can be powered by any of the following sources:

- External Power source (3.6 to 7V)
- USB +5V
- Lithium-Ion battery

At least one of the above power sources should be connected to the ezLCD-002.

Power ON/OFF

When any of the above power sources is connected, the ezLCD-002 may be **powered on** by:

- applying a positive voltage (2V to 7V) to the ON/OFF pin for at least 1 ms or
- by pressing the ON button (Figure 6) on the back of the display

The ezLCD-002 may be **powered off** by:

- connecting ON/OFF pin to the ground for at least 1 ms or
- by pressing the OFF button (Figure 6) on the back of the display
- The OFF button has the priority over the ON button.

When the ON/OFF/ pin is jumpered to the power source, the ezLCD power will be cycled by connecting and disconnecting the power source.

Outputs

The ezLCD-002 Power Supply drives the following outputs:

- +3V Main Power Vcc (MPU, SED1375, Interfaces)
- +3.6 to +3.8V LCD (LCD Screen V_1)
- +3V LCD (LCD Screen V_2)
- 16mA constant current (LCD Light)
- Lithium-Ion battery charge current

NOTE: The +3V Ref is an I/O reference voltage.

It may be used as a pull-up source (I2C etc.).
It SHOULD NOT be used as a power source.

Battery Charging

When the ezLCD-002 is powered by USB only, the charge current is set to 100mA (max). When External Power is connected, charging current is set at 280mA (typ).
The ezLCD embedded battery charger features a precharge current to protect deeply discharged cells. If battery voltage is less than 3V, the device enters a precharge mode where charging current is limited to 40mA.

LCD Light

The LCD Light is powered by 16mA generated by the White LED Driver. The LCD Light can be turned on or off by the LIGHT_ON signal from MPU (ezLCD commands: LIGHT_ON and LIGHT_OFF). Additionally, the light can be turned on by jumping the signal LIGHT_ON# to the GND on the CN1 or CN3 connector. The Light On condition has priority over Light Off. For example, once LIGHT_ON# is jumpered to the GND, the light cannot be extinguished by sending the LIGHT_OFF command to the ezLCD-002. The following table shows the LCD Light logic.

LIGHT_ON	LIGHT_ON#	LCD Light
OFF	Open	Off
OFF	GND	On
ON	Open	On
ON	GND	On

2.3.4 RS-232

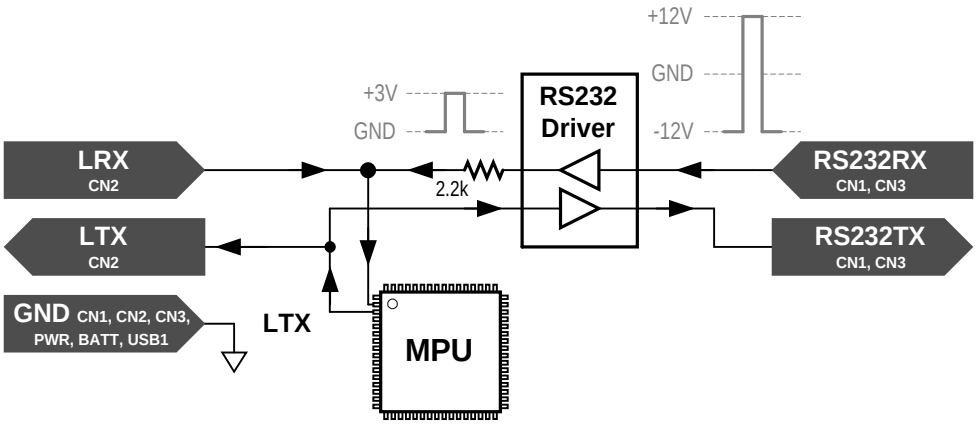


Figure 7. ezLCD-002 RS232 Interface

Default Communication Parameters

Baudrate: ..... 115200 bps
No of Stop Bits: 1
Parity: ..... Off
Handshake: ..... None

Operation

RS232:
The ezLCD-002 uses 3 wires for a non-handshake RS232 communication:

- RS232 RX (ezLCD receive)
- RS232 TX (ezLCD transmit)
- GND (common ground)

The voltage levels and limits are as per RS232 standard.
The MPU handles the asynchronous communication protocol. The RS232 Driver converts voltage levels from MPU 0V(Lo) and 3V(Hi) to RS232 -12V(Lo) and +12V(Hi).

Asynchronous Serial:
The ezLCD-002 uses 3 wires for a non-handshake
Asynchronous Serial (RS232-TTL) communication:

- LRX (ezLCD receive)
- LTX (ezLCD transmit)
- GND (common ground)

The voltage levels are:

- 0V to +1V = Lo (logical "0")
- +2V to +3V = Hi (logical "1")
- Absolute minimum: -0.2V
- Absolute maximum: +3.2V

The MPU handles the asynchronous communication protocol. The Asynchronous Serial Interface uses the same MPU lines as the RS232 does. The 2.2k resistor is used to separate the receive signals from both interfaces. The Asynchronous Serial receive has the priority over the RS232 receive

2.3.5 USB

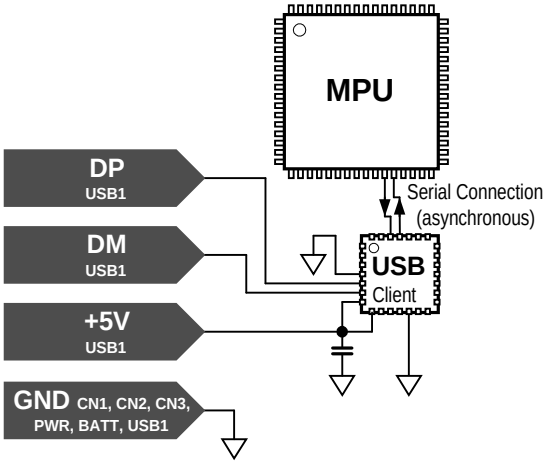


Figure 8. ezLCD-002 USB Interface

Operation

Connector CN1

The ezLCD-002 USB Interface uses 4 lines:

- DM (USB Data Minus)
- DP (USB Data Plus)
- +5V
- GND

USB Client IC

The USB Client IC (CP2101 by Silicon Laboratories Inc.), handles all protocol and physical layer aspects of the USB communication.
MPU communicates with the USB Client through standard asynchronous serial connection using the following communication parameters:

- Baudrate: ..... 115200 bps
- No. of Stop Bits: 1
- Parity: ..... Off

Host Configuration

The ezLCD CD includes ready-to-go, royalty free USB drivers, configure the operating system of the Host Computer to "see" the ezLCD-002 as an additional COM port.
When ezLCD-002 USB is configured as a COM port, the following communication parameters should be used:
Baudrate: ..... 115200 bps
No. of Stop Bits: 1
Parity: ..... Off
Handshake: ..... None

Drivers , Software and Documentation

The ezLCD CD includes the CP2101 USB drivers.
The latest documentation, software and drivers are provided by the Silicon Laboratories Inc.

2.3.6 I2C

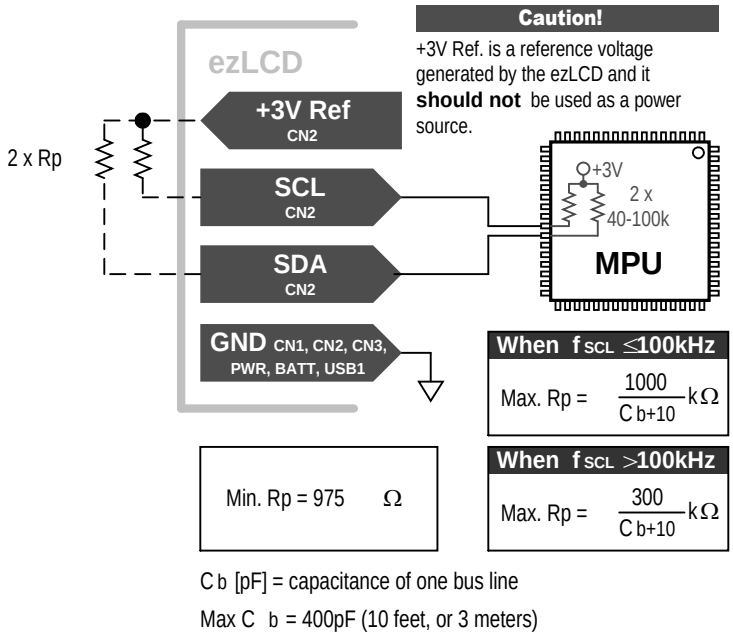

Caution!
+3V Ref. is a reference voltage generated by the ezLCD and it should not be used as a power source.
ezLCD
+3V Ref
CN2
SCL
CN2
SDA
CN2
GND CN1, CN2, CN3, PWR, BATT, USB1
2 x Rp
+3V
2 x 40-100k
MPU
When f SCL ≤100kHz
Max. Rp = 1000 / C b+10 kΩ
When f SCL >100kHz
Max. Rp = 300 / C b+10 kΩ
Min. Rp = 975 Ω
C b [pF] = capacitance of one bus line
Max C b = 400pF (10 feet, or 3 meters)

Figure 9. ezLCD-002 I2C Interface

Operation

Connector CN2

The ezLCD-002 I2C Interface uses 3 wires:

- SCL (Clock)
- SDA (Data)
- GND

Pull-Up Resistors

The pull-up resistors (Rp) should be connected to +3V.
The ezLCD-002 outputs +3V reference voltage, which may be used as a pull-up source, as it is shown on the *Figure 9.* above.

Protocol

- Configuration:
  The ezLCD-002 is configured as an I2C Slave.
- Address:
  The default I2C address of the ezLCD-002 is 111 dec (6F hex).
- Handshake:
  The ezLCD-002 responds with NACK (non-acknowledge) if its' 1024 byte command circular buffer runs out of space.

Reminder:

I2C address byte consists of the 7 address bits and the R/W bit in LSB.
This means that the address byte should be:

- 222 dec (DE hex) when sending commands to the ezLCD
- 223 dec (DF hex) when reading the Touch Screen data

2.3.7 Touch Screen

	0	1	2	3	4	5	6	7
0	$00	$01	$02	$03	$04	$05	$06	$07
1	$10	$11	$12	$13	$14	$15	$16	$17
2	$20	$21	$22	$23	$24	$25	$26	$27
3	$30	$31	$32	$33	$34	$35	$36	$37
4	$40	$41	$42	$43	$44	$45	$46	$47
5	$50	$51	$52	$53	$54	$55	$56	$57

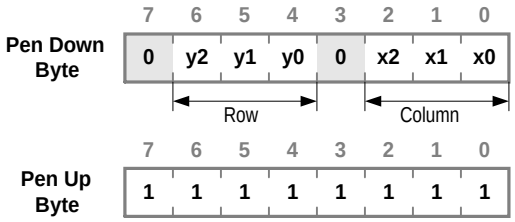


Figure 10. ezLCD-002 Touch Screen Organization

Operation

The Touch Screen data is transmitted by the ezLCD through RS-232, USB and I2C interfaces.
The ezLCD-002 Touch Screen is divided into 5 rows and 7 columns.
The Touch Screen operation is event-driven.
When the Touch Screen is pressed, the ezLCD sends the **Pen Down Byte**, which contains the coordinates of the pressed area. The format of the **Pen Down Byte** is shown on the drawing above.
The **Pen Down Byte** is retransmitted if the coordinates have been changed while the screen is pressed.
When the Touch Screen is released, the ezLCD sends the **Pen Up Byte** (FF hex).

RS-232 and USB

Touch Screen data transmission is triggered by the Touch Screen event (see above).

I2C

Since:
1. The I2C communication have to be initiated by the Master and
2. ezLCD-002 is configured as an I2C Slave,

the ezLCD-002 Touch Screen data transmission is triggered by the I2C read operation when the ezLCD-002 is addressed (Ref: I2C). Only the last recorded Touch Screen byte is sent (Pen Down or Pen Up byte). Master should be periodically polling ezLCD-002 for the new data.

How to read the touch using I2C

1. Master: Sends START then ezLCD address byte with R/W bit set: 111*2+1 = 223 (DF hex).
2. Slave (ezLCD): Responds with ACK and sends the last touch screen data (Pen Down or Pen Up byte).
3. Master: Sends ACK (or NACK) and STOP.

Notes:
ezLCD sends always 1 byte per START/STOP
ezLCD switches to "Not Addressed Slave" mode after sends the data and receives ACK or NACK (see 3. above).

2.4 Firmware

2.4.1 Flash Memory

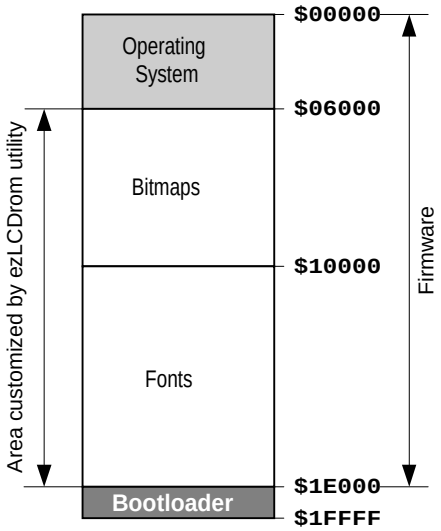


Figure 11. ezLCD-002 Flash Memory Map

The non-volatile memory (flash, ROM) of the ezLCD-001 is divided into 4 segments:
1. Operating System
2. Bitmaps
3. Fonts
4. Bootloader

The ezLCD-002 firmware upgrade file consists of Operating System, Bitmaps and Fonts segments. Out of those Bitmaps and Fonts can be user-modified by the ezLCDrom utility.

Software in the Bootloader segment is used for the firmware upgrades and customization. The Bootloader permanently resides in the ezLCD-002 flash. It is not affected by the firmware upgrades.

2.4.1.1 Operating System

Start Address: 00000 hex
End Address: 05FFF hex
Size: 24kB
Upgradable: Yes
Customizable: No

The Operating System segment holds all of the ezLCD-002 operational software:

- ezLCD Command Processor
- I/O routines for all the interfaces
- LCD control
- Graphic routines

The Operating System segment is modified by each Firmware Upgrade (segment is upgradable). This segment cannot be customized by the ezLCDrom utility (segment is not customizable).

2.4.1.2 Bitmaps

Start Address: 06000 hex
End Address: 0FFFF hex
Size: 40kB
Upgradable: Yes
Customizable: Yes

The Bitmaps segment is used for the storage of the user bitmaps (icons). The bitmaps are stored in compressed form by the ezLCDrom utility. They can be displayed on the screen by using the PUT_ICON command.

The Bitmaps segment is modified by each Firmware Upgrade (segment is upgradable).
This segment can be customized by the ezLCDrom utility (segment is customizable).

Note: The contents of this segment is overwritten by each Firmware Upgrade. Customization of the firmware file (by the ezLCDrom utility) should be done before the actual ezLCD-002 upload.

2.4.1.3 Fonts

Start Address: 10000 hex
End Address: 1DFFF hex
Size: 56kB
Upgradable: Yes
Customizable: Yes

The Fonts segment is used to store the screen fonts of the ezLCD-002.
The following default fonts are implemented in the firmware:

Font 0: ezLCD
Font 1: ezLCD
Font 2: ezLCD
Font 3: ezLCD
Font 4: ezLCD
Font 5: ezLCD

The above fonts can be modified or replaced by the ezLCDrom utility.

The Fonts segment is modified by each Firmware Upgrade (segment is upgradable).
This segment can be customized by the ezLCDrom utility (segment is customizable).

Note: The contents of this segment is overwritten by each Firmware Upgrade. Customization of the firmware file (by the ezLCDrom utility) should be done before the actual ezLCD-002 upload.

2.4.1.4 Bootloader

Start Address: 1E000 hex
End Address: 1FFFF hex
Size: 8kB
Upgradable: No
Customizable: No

Software in the Bootloader segment is used for firmware upgrades and customization. The Bootloader permanently resides in the ezLCD-002 flash. It is not affected by firmware upgrades. More information about the Bootloader can be found in the Firmware Upgrade and ezLCDrom chapters.

The Bootloader segment cannot be modified by the Firmware Upgrade (segment is not upgradable). This segment cannot be customized by the ezLCDrom utility (segment is not customizable).

2.4.2 Firmware Upgrade

The firmware upload to the ezLCD-002 can be done through either the RS232 or USB interface. EzLCD-002 must be running the Bootloader software. The Bootloader always resides in the highest segment of the ezLCD-002 flash memory. The program will jump into the Bootloader when the PGR button on the back of the ezLCD-002 is pressed. Once ezLCD-002 software enters the Bootloader, it stays there until the next power-up/reset.

The firmware can be uploaded to the ezLCD-002 by the ezLCDrom utility. Since the Bootloader responds to the Atmel's STK500 commands, the firmware can also be uploaded by the Atmel AVR Studio available at: http://www.atmel.com/products/AVR/ in "Tools and Software" section.

To upgrade ezLCD-002 firmware:

1. Power off the ezLCD-002
2. Connect PC to the ezLCD-002 via USB or RS232 interface
3. Press and hold down PGR button on back of ezLCD-002
4. Power on the ezLCD-002
5. Release the PGR Button
6. Upload the firmware using the ezLCDrom utility or Atmel AVR Studio
7. Power off the ezLCD-002
8. Power on the ezLCD-002

2.5 ezLCD Commands

The ezLCD Commands may be fed to the ezLCD through any of the available interfaces.

General
CLS
LIGHT_ON
LIGHT_OFF
SET_COLOR
SET_XY

Points
PLOT
PLOT_XY

Lines
H_LINE
V_LINE
LINE_TO_XY

Figures
ARC
CIRCLE_R
CIRCLE_R_FILL
BOX
BOX_FILL

Bitmaps
PUT_BITMAP
PUT_ICON
PUT_SF_ICON
PICTURE

Text and Fonts
SELECT_FONT
SET_BG_COLOR
TEXT_NORTH
TEXT_EAST
TEXT_SOUTH
TEXT_WEST
PRINT_CHAR
PRINT_CHAR_BG
PRINT_STRING
PRINT_STRING_BG

2.5.1 ARC

Description: Draws an Arc in Current Color, with the center at Current Position, starting on Begin Angle and ending on the End Angle.
Class: Multi Byte Command
Code: **2F**hex, **47**dec, **/** ASCII

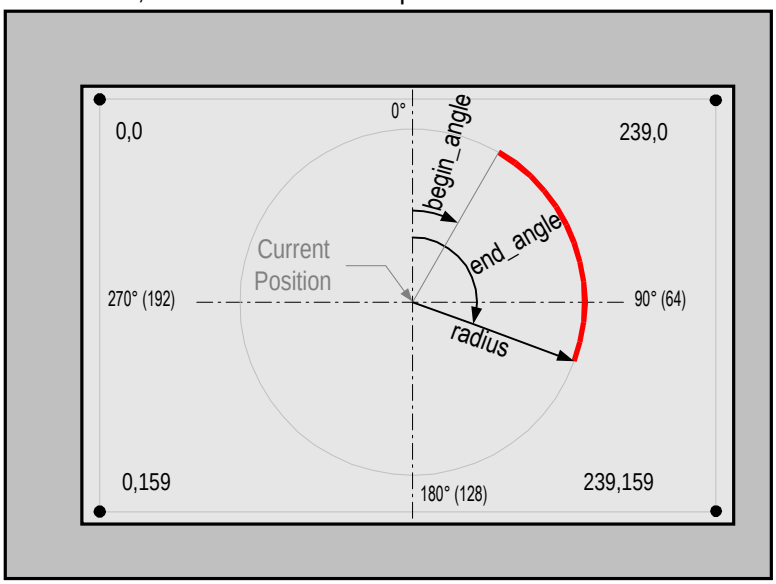
7	6	5	4	3	2	1	0	
ARC								**Byte 0 (Command)**
r a d i u s								**Byte 1 (Radius)**
begin_angle								**Byte 2 (Arc Begin Angle)**
end_angle								**Byte 3 (Arc End Angle)**

See Also: SET_XY, SET_COLOR, CIRCLE_R

Angle Coding: The angle range is from 0 to 255.
To transform degrees to ARC angle units:
`Angle_lcd = Angle_deg x 32 / 45`
For example:

```
 32 =  45°
 64 =  90°
128 = 180°
192 = 270°
  0 = 0° = 360°
```

The angle is drawn clockwise with the zero positioned at the top of a screen, as it is shown on the picture below



Example:

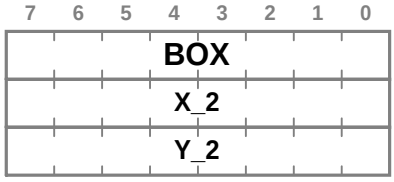
The following sequence will draw a green arc from 45 to 225 degrees with the center positioned in the middle of a screen.

```
SET_COLOR       24 hex
```

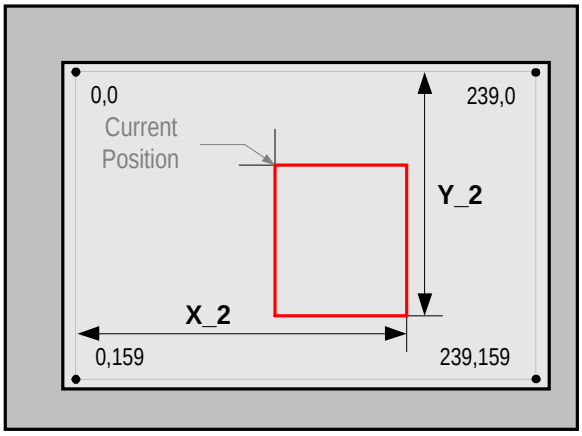
```
GREEN          00111000 bin
SET_XY          25 hex
120            120 dec
 80             80 dec
ARC             2F hex
 60             60 dec (radius)
 32             32 dec (begin_angle = 45 degrees)
160            160 dec (end_angle = 225 degrees)
```

2.5.2 BOX

Description: Draws a rectangle.
Class: Multi Byte Command
Code: **42**hex, **66**dec, **B** ASCII



Byte 0 (Command)
Byte 1 (Corner Column)
Byte 2 (Corner Row)



See Also: SET_XY, BOX_FILL

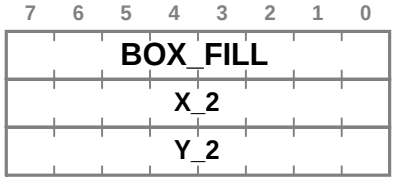
Example:

The following sequence will draw the red rectangle

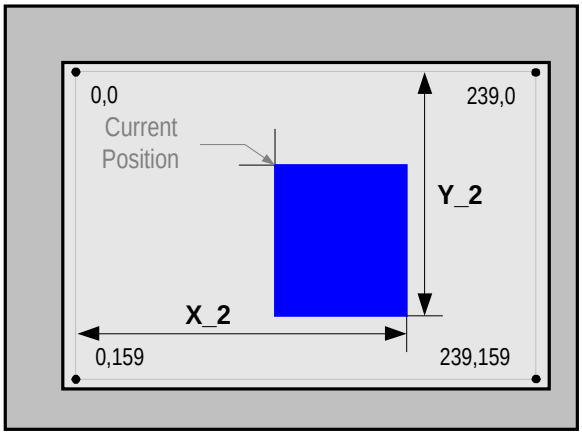
```
SET_COLOR    24 hex
RED        00000111 bin
SET_XY       25 hex
 95          95 dec
 40          10 dec
BOX          42 hex
180         180 dec (X_2)
120         120 dec (Y_2)
```

2.5.3 BOX_FILL

Description: Draws a rectangle filled with Current Color
Class: Multi Byte Command
Code: **43**hex, **67**dec, **C** ASCII



Byte 0 (Command)
Byte 1 (Corner Column)
Byte 2 (Corner Row)



See Also: SET_XY, BOX

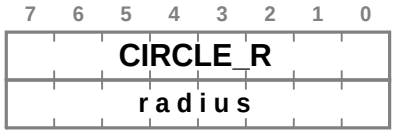
Example:

The following sequence will draw the rectangle filled with blue color

```
SET_COLOR    24 hex
RED        11000000 bin
SET_XY       25 hex
 95          95 dec
 40          10 dec
BOX_FILL     43 hex
180         180 dec (X_2)
120         120 dec (Y_2)
```

2.5.4 CIRCLE_R

Description: Draws a circle in Current Color at Current Position
Class: Double Byte Command
Code: **29**hex, **41**dec, **)** ASCII



Byte 0 (Command)

Byte 1 (radius)

See Also: SET_XY, SET_COLOR

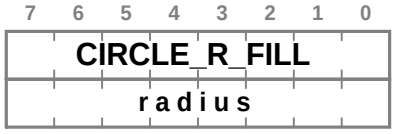
Example:

The following sequence will draw a green circle in the middle of the screen.

```
SET_COLOR    24 hex
GREEN        00111000 bin
SET_XY       25 hex
120         120 dec
80           80 dec
CIRCLE_R     29 hex
60           60 dec
```

2.5.5 CIRCLE_R_FILL

Description: Draws a circle in Current Color at Current Position, filled with Current Color
Class: Double Byte Command
Code: **39**hex, **57**dec, **9** ASCII



Byte 0 (Command)

Byte 1 (Radius)

See Also: SET_XY, SET_COLOR

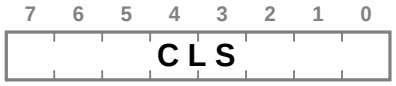
Example:

The following sequence will draw a red filled circle in the middle of the screen.

```
\SET_COLOR       24 hex
RED             00000111 bin
SET_XY           25 hex
120             120 dec
80               80 dec
CIRCLE_R_FILL    39 hex
60               60 dec
```

2.5.6 CLS

Description: Clears screen by filling it with the Current Color
Class: Single Byte Command
Code: **21**hex, **33**dec, **!** ASCII



Byte 0 (Command)

See Also: SET_COLOR

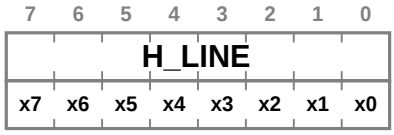
Example:

The following sequence will clear the screen

```
SET_COLOR   24 hex
WHITE       11111111 bin
CLS         21 hex
```

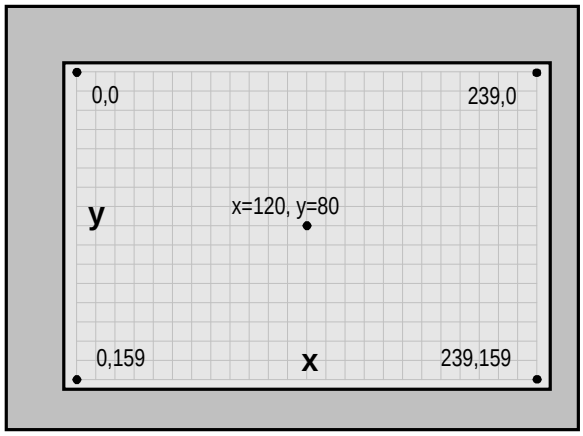
2.5.7 H_LINE

Description: Fast draws a horizontal line from Current Position, to the column specified by the parameter.
Class: Double Byte Command
Code: **40**hex, **64**dec, @ ASCII



Byte 0 (Command)
Byte 1 (X)

Note: The screen size is 240x160. However, the valid X range is 0 - 255



See Also: V_LINE, SET_XY

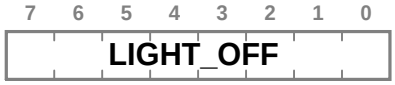
Example:

The following sequence will draw the horizontal green line from (20, 60) to (170, 60)

```
SET_COLOR    24 hex
GREEN      00111000 bin
SET_XY       25 hex
 20          20 dec
 60          60 dec
H_LINE       40 hex
170         170 dec
```

2.5.8 LIGHT_OFF

Description: Turns off the screen light
Class: Single Byte Command
Code: **23**hex, **35**dec, # ASCII



Byte 0 (Command)

See Also: LIGHT_ON

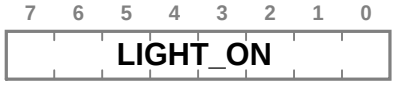
Example:

The following sequence will turn off the screen light

```
LIGHT_OFF   23 hex
```

2.5.9 LIGHT_ON

Description: Turns on the screen light
Class: Single Byte Command
Code: **22**hex, **34**dec, **"** ASCII



Byte 0 (Command)

See Also: LIGHT_OFF

Example:

The following sequence will turn on the screen light

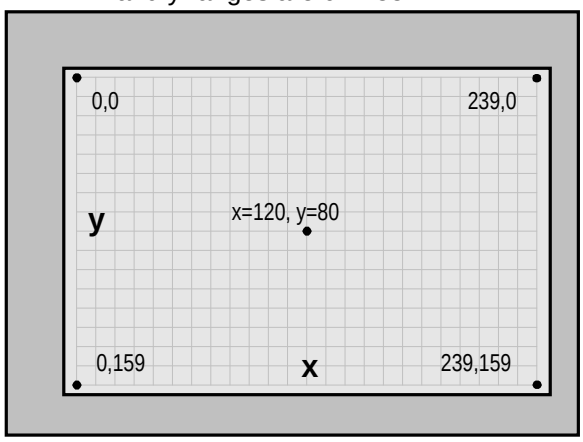
```
LIGHT_ON    22 hex
```

2.5.10 LINE_TO_XY

Description: Draws a line in Current Color, from the Current Position to the to specified position
Class: Multi Byte Command
Code: **28**hex, **40**dec, **(** ASCII

7	6	5	4	3	2	1	0	
LINE_TO_XY								**Byte 0 (Command)**
x7	x6	x5	x4	x3	x2	x1	x0	**Byte 1 (x)**
y7	y6	y5	y4	y3	y2	y1	y0	**Byte 2 (y)**

Note: The screen size is 240x160. However, the valid x and y ranges are 0 - 255



See Also: SET_XY, SET_COLOR, PLOT

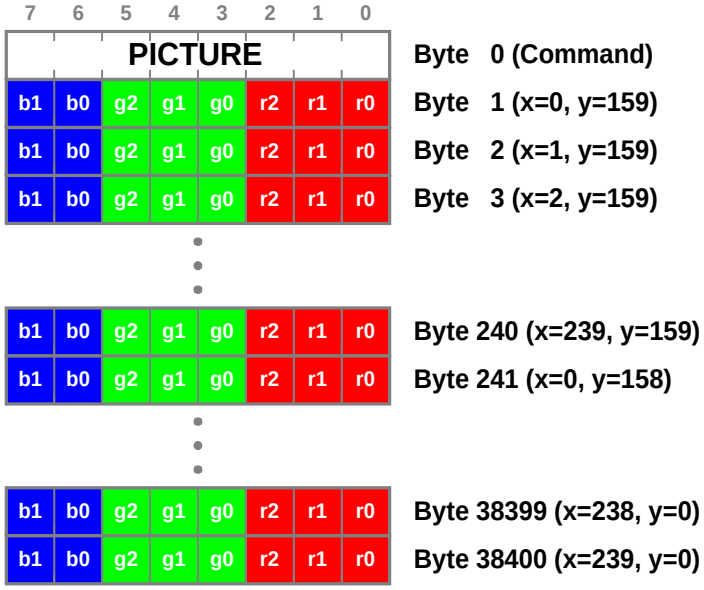
Example:

The following sequence will draw a red line across the screen.

```
SET_COLOR     24 hex
RED           00000111 bin
SET_XY        25 hex
0              0 dec
0              0 dec
LINE_TO_XY    28 hex
239          239 dec
159          159 dec
```

2.5.11 PICTURE

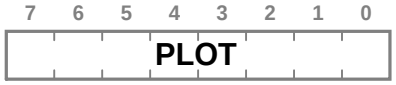
Description: Puts a bitmap picture over the entire screen
Class: Multi Byte Command
Code: **2A**hex, **42**dec, ***** ASCII


7 6 5 4 3 2 1 0
PICTURE
b1 b0 g2 g1 g0 r2 r1 r0
Byte 0 (Command)
Byte 1 (x=0, y=159)
Byte 2 (x=1, y=159)
Byte 3 (x=2, y=159)
Byte 240 (x=239, y=159)
Byte 241 (x=0, y=158)
Byte 38399 (x=238, y=0)
Byte 38400 (x=239, y=0)

See Also: SET_XY, SET_COLOR, PUT_BITMAP

2.5.12 PLOT

Description: Plots a point at Current Position in Current Color
Class: Single Byte Command
Code: **26**hex, **38**dec, **&** ASCII



Byte 0 (Command)

See Also: SET_XY, SET_COLOR

Example:

The following sequence will put the blue point in the middle of the screen.

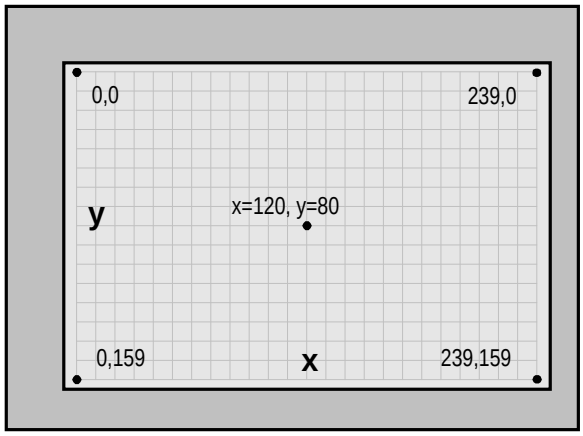
```
SET_COLOR    24 hex
BLUE         11000000 bin
SET_XY       25 hex
120         120 dec
80           80 dec
PLOT         26 hex
```

2.5.13 PLOT_XY

Description: Plots a point in Current Color, at specified position.
Class: Multi Byte Command
Code: **27**hex, **39**dec, **'** ASCII

7	6	5	4	3	2	1	0	
PLOT_XY								**Byte 0 (Command)**
x7	x6	x5	x4	x3	x2	x1	x0	**Byte 1 (x)**
y7	y6	y5	y4	y3	y2	y1	y0	**Byte 2 (y)**

Note: The screen size is 240x160. However, the valid x and y ranges are 0 - 255



See Also: SET_XY, SET_COLOR, PLOT

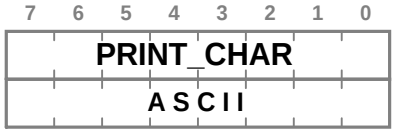
Example:

The following sequence will put the red point in the middle of the screen.

```
SET_COLOR   24 hex
RED         00000111 bin
PLOT_XY     27 hex
120         120 dec
80          80 dec
```

2.5.14 PRINT_CHAR

Description: Prints a character at Current Position
Class: Double Byte Command
Code: **2C**hex, **44**dec, **,** ASCII



Byte 0 (Command)

Byte 1 (ASCII Character)

See Also: SELECT_FONT, PRINT_STRING

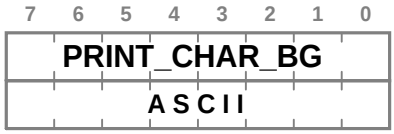
Example:

The following sequence will print black character 'M' in the middle of the screen, using font number 2

```
SELECT_FONT    2B hex
2               2 dec
SET_COLOR      24 hex
BLACK    00000000 bin
SET_XY         25 hex
120           120 dec
80             80 dec
PRINT_CHAR     2C hex
'M'            4D hex
```

2.5.15 PRINT_CHAR_BG

Description: Prints a character at Current Position on the background specified by SET_BG_COLOR command
Class: Double Byte Command
Code: **3C**hex, **60**dec, **<** ASCII



Byte 0 (Command)

Byte 1 (ASCII Character)

See Also: SELECT_FONT, SET_BG_COLOR, PRINT_STRING_BG

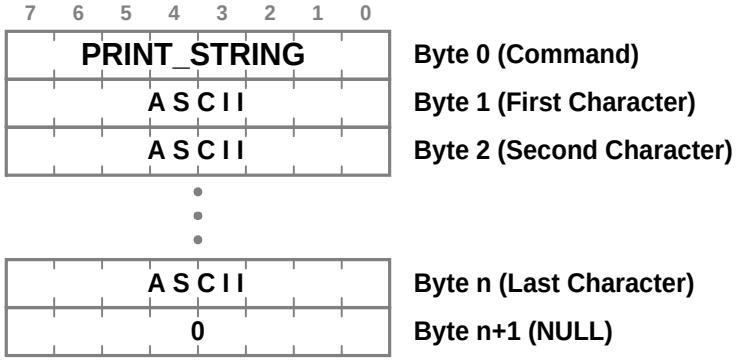
Example:

The following sequence will print white character 'M', on a black background in the middle of the screen, using font number 2

```
SELECT_FONT   2B hex
2              2 dec
SET_BG_COLOR  34 hex
BLACK          00000000 bin
SET_COLOR     24 hex
WHITE          11111111 bin
SET_XY        25 hex
120          120 dec
80            80 dec
PRINT_CHAR_BG 3C hex
'M'           4D hex
```

2.5.16 PRINT_STRING

Description: Prints null-terminated String starting at Current Position
Class: Multi Byte Command
Code: **2D**hex, **45**dec, **-** ASCII



See Also: SELECT_FONT, PRINT_CHAR

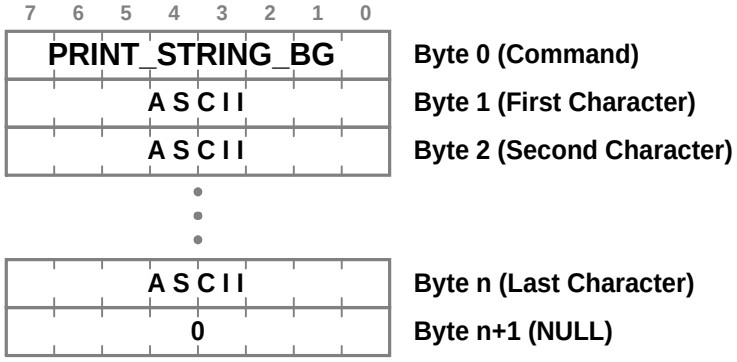
Example:

The following sequence will print violet sign "LCD" in the middle of the screen, using font number 1

```
SELECT_FONT   2B hex
1              1 dec
SET_COLOR     24 hex
VIOLET        11000100 bin
SET_XY        25 hex
120          120 dec
80            80 dec
PRINT_STRING  2D hex
'L'           4C hex
'C'           43 hex
'D'           44 hex
NULL           0 hex
```

2.5.17 PRINT_STRING_BG

Description: Prints null-terminated String starting at Current Position on the background specified by SET_BG_COLOR command
Class: Multi Byte Command
Code: **3D**hex, **61**dec, **=** ASCII



See Also: SELECT_FONT, SET_BG_COLOR, PRINT_CHAR_BG

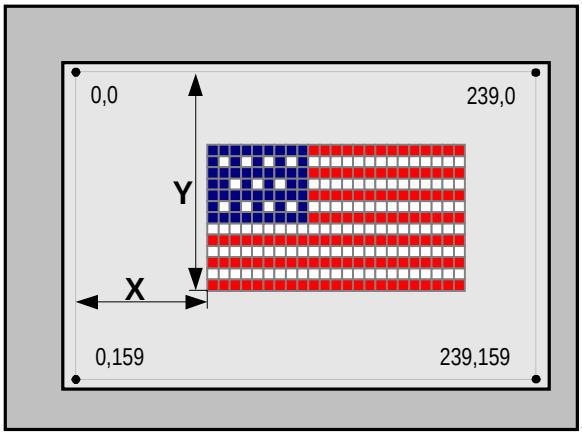
Example:

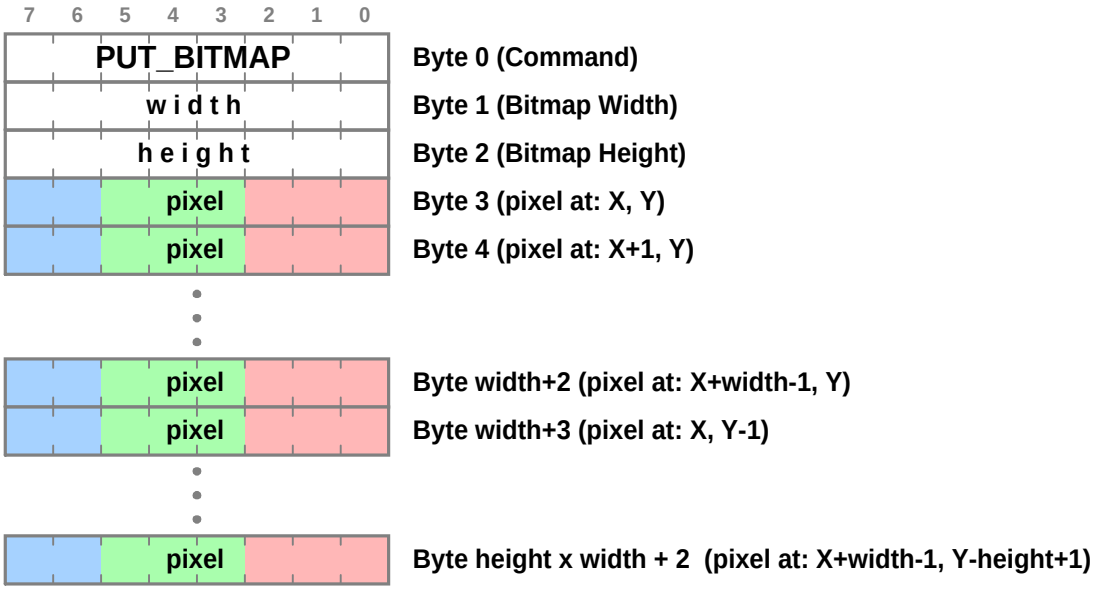
The following sequence print Yellow "LCD" on the Navy background, in the middle of a screen, using font no 0.

```
SET_BG_COLOR     34 hex
NAVY             10000000 bin
SET_COLOR        24 hex
YELLOW           00111111 bin
SET_XY           25 hex
120             120 dec
80               80 dec
SELECT_FONT      2B hex
0                 0 dec
PRINT_STRING_BG  3D hex
'L'              4C hex
'C'              43 hex
'D'              44 hex
NULL              0 hex
```

2.5.18 PUT_BITMAP

Description: Puts Bitmap on the screen starting at Current Position, then UP and RIGHT
Class: Multi Byte Command
Code: **2E**hex, **46**dec, **.** ASCII




7 6 5 4 3 2 1 0
PUT_BITMAP
w i d t h
h e i g h t
pixel
pixel
pixel
pixel
pixel
Byte 0 (Command)
Byte 1 (Bitmap Width)
Byte 2 (Bitmap Height)
Byte 3 (pixel at: X, Y)
Byte 4 (pixel at: X+1, Y)
Byte width+2 (pixel at: X+width-1, Y)
Byte width+3 (pixel at: X, Y-1)
Byte height x width + 2 (pixel at: X+width-1, Y-height+1)

Note: The total number of bytes is: width x height + 3

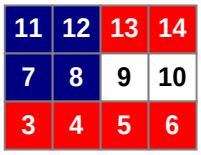
See Also: SET_XY, SET_COLOR, PICTURE

Example:

The following sequence will put 4x3 bitmap at x = 60, y = 80

```
SET_XY        25 hex
x             60 dec
y             80 dec
PUT_BITMAP    2E hex    ----------+--
width          4 dec              |
height         3 dec              |
```

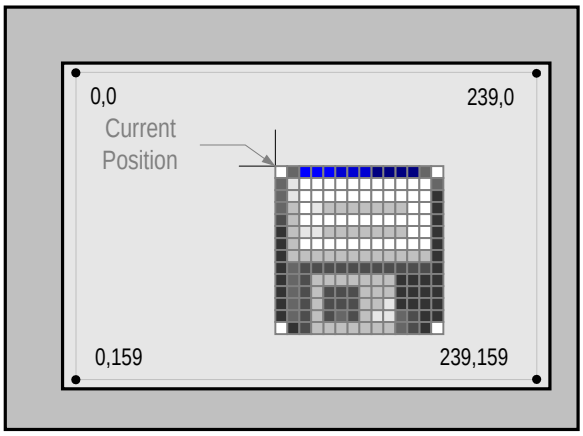
```
pixel (x = 60, y = 80)            |
pixel (x = 61, y = 80)            |
pixel (x = 62, y = 80)            |
pixel (x = 63, y = 80)          TOTAL:
pixel (x = 60, y = 79)     4 x 3 + 3 = 15 bytes
pixel (x = 61, y = 79)            |
pixel (x = 62, y = 79)            |
pixel (x = 63, y = 79)            |
pixel (x = 60, y = 78)            |
pixel (x = 61, y = 78)            |
pixel (x = 62, y = 78)            |
pixel (x = 63, y = 78)  ----------+
```

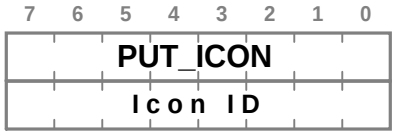


2.5.19 PUT_ICON

Description: Displays the icon with it's upper-left corner positioned at the Current Position.
The icon is read from the ezLCD ROM.
Use the ezLCDrom utility to store icons in the ezLCD ROM

Class: Double Byte Command
Code: **57**hex, **87**dec, **W** ASCII





Byte 0 (Command)

Byte 1 (Icon ID)

See Also: SET_XY

Example:

The following sequence will display an icon no 3 with it's upper-left corner positioned at X = 60, Y = 43

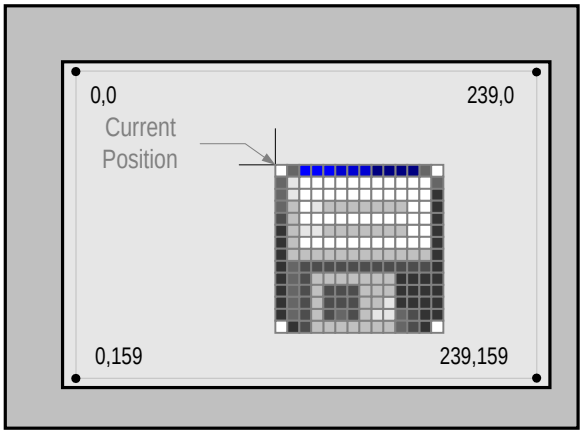
```
SET_XY      25 hex
60          60 dec
43          43 dec
PUT_ICON    57 hex
3            3 dec
```

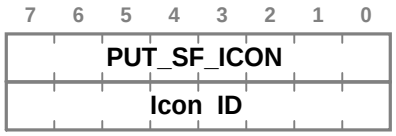
2.5.20 PUT_SF_ICON

Description: Displays the serial flash icon with it's upper-left corner positioned at the Current Position.

The icon is read from the ezLCD-002 1Mbyte Serial Flash.
Use the ezLCDflash utility to store icons in the ezLCD Serial Flash

Class: Double Byte Command
Code: **58**hex, **87**dec, **X** ASCII





Byte 0 (Command)

Byte 1 (Icon ID: 0 to 254)

See Also: SET_XY

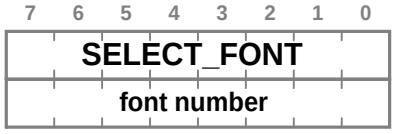
Example:

The following sequence will display a serial flash icon no 176 with it's upper-left corner positioned at X = 60, Y = 43

```
SET_XY          25 hex
60              60 dec
43              43 dec
PUT_SF_ICON     58 hex
176            176 dec
```

2.5.21 SELECT_FONT

Description: Sets the Current Font
Class: Double Byte Command
Code: **2B**hex, **43**dec, **+** ASCII



Byte 0 (Command)

Byte 1 (font number)

Note: The following fonts are implemented

Font 0: ezLCD
Font 1: ezLCD
Font 2: ezLCD
Font 3: ezLCD
Font 4: ezLCD
Font 5: ezLCD

See Also: PRINT_STRING, PRINT_CHAR

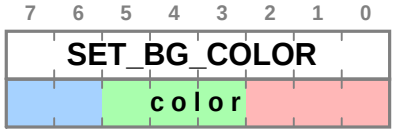
Example:

The following sequence will print black character 'M' in the middle of the screen, using font number 2

```
SELECT_FONT    2B hex
2               2 dec
SET_COLOR      24 hex
BLACK    00000000 bin
SET_XY         25 hex
120           120 dec
80             80 dec
PRINT_CHAR     2C hex
'M'            4D hex
```

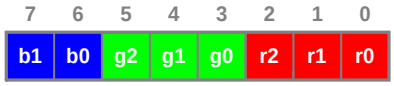
2.5.22 SET_BG_COLOR

Description: Sets the Background Color for the following instructions:
PRINT_CHAR_BG
PRINT_STRING_BG
Class: Double Byte Command
Code: **34**hex, **52**dec, **4** ASCII



Byte 0 (Command)
Byte 1 (Color Code)

Note: The default NATURAL palette has the following color coding:



See Also: PRINT_CHAR_BG, PRINT_STRING_BG, PALETTE

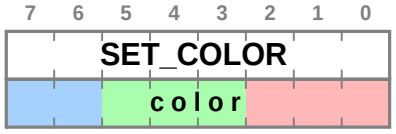
Example:

The following sequence print Yellow "LCD" on the Navy background, in the middle of a screen, using font no 0.

```
SET_BG_COLOR     34 hex
NAVY             10000000 bin
SET_COLOR        24 hex
YELLOW           00111111 bin
SET_XY           25 hex
120             120 dec
80               80 dec
SELECT_FONT      2B hex
0                 0 dec
PRINT_STRING_BG  3D hex
'L'              4C hex
'C'              43 hex
'D'              44 hex
NULL              0 hex
```

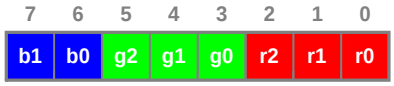
2.5.23 SET_COLOR

Description: Sets the Current Color
Class: Double Byte Command
Code: **24**hex, **36**dec, **$** ASCII



Byte 0 (Command)
Byte 1 (Color Code)

Note: The default NATURAL palette has the following color coding:



See Also: CLS, PLOT, PALETTE

Example:

The following sequence will fill the whole display with green

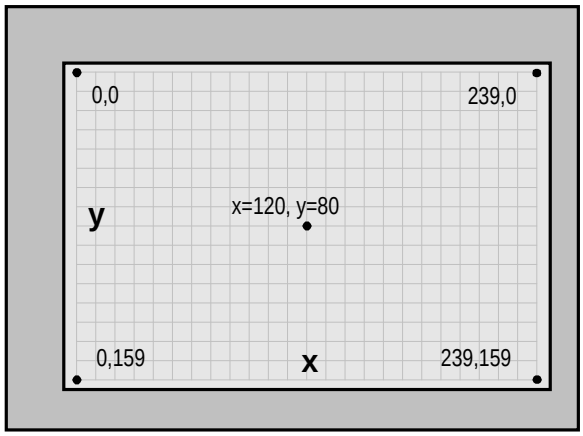
```
SET_COLOR    24 hex
GREEN        00111000 bin
CLS          21 hex
```

2.5.24 SET_XY

Description: Sets the Current Position
Class: Multi Byte Command
Code: **25**hex, **37**dec, **%** ASCII

7	6	5	4	3	2	1	0	
SET_XY								**Byte 0 (Command)**
x7	x6	x5	x4	x3	x2	x1	x0	**Byte 1 (x)**
y7	y6	y5	y4	y3	y2	y1	y0	**Byte 2 (y)**

Note: The screen size is 240x160. However, the valid x and y ranges are 0 - 255



See Also: PLOT, LINE_TO_XY, CIRCLE_R

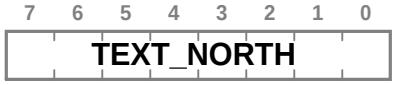
Example:

The following sequence will put the blue point in the middle of the screen.

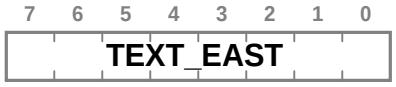
```
SET_COLOR    24 hex
BLUE         11000000 bin
SET_XY       25 hex
120         120 dec
80           80 dec
PLOT         26 hex
```

2.5.25 TEXT_EAST

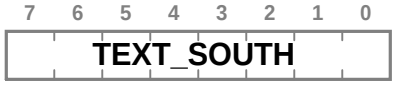
Description: Set the orientation of the text, as shown on the picture below
Class: Single Byte Commands
Code: **TEXT_NORTH**: **60**hex, **96**dec, **'** ASCII
TEXT_EAST : **61**hex, **97**dec, **a** ASCII
TEXT_SOUTH: **62**hex, **98**dec, **b** ASCII
TEXT_WEST : **2F**hex, **99**dec, **c** ASCII



Byte 0 (Command)



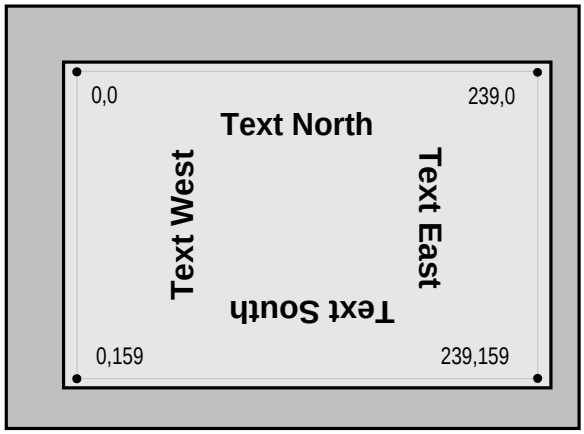
Byte 0 (Command)



Byte 0 (Command)

Byte 0 (Command)

Note: TEXT_NORTH is the default text orientation



See Also: PRINT_CHAR, PRINT_STRING, SELECT_FONT

Example:

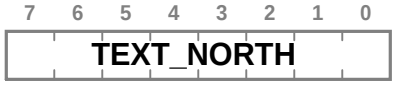
The following sequence will print the text pattern similar to the one on the picture above.

```
SET_XY         25 hex
60             60 dec
10             10 dec
SELECT_FONT    2B hex
0               0 dec
TEXT_NORTH     60 hex
PRINT_STRING   2D hex
"Text North "
```

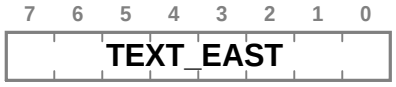
```
NULL            0 hex
TEXT_EAST      61 hex
PRINT_STRING   2D hex
"  Text East  "
NULL            0 hex
TEXT_SOUTH     62 hex
PRINT_STRING   2D hex
" Text South "
NULL            0 hex
TEXT_WEST      63 hex
PRINT_STRING   2D hex
"  Text West "
NULL            0 hex
```

2.5.26 TEXT_NORTH

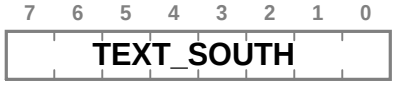
Description: Set the orientation of the text, as shown on the picture below
Class: Single Byte Commands
Code: **TEXT_NORTH**: **60**hex, **96**dec, **'** ASCII
TEXT_EAST : **61**hex, **97**dec, **a** ASCII
TEXT_SOUTH: **62**hex, **98**dec, **b** ASCII
TEXT_WEST : **2F**hex, **99**dec, **c** ASCII



Byte 0 (Command)



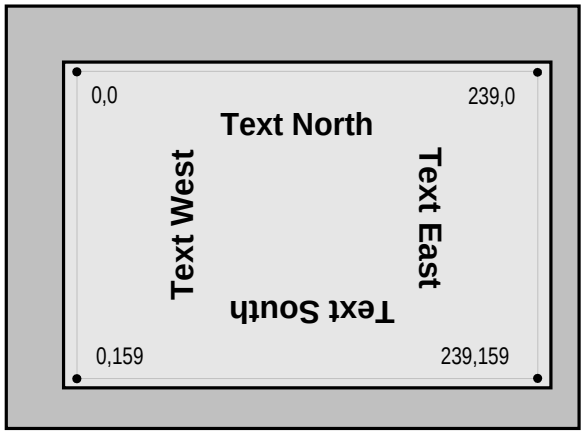
Byte 0 (Command)



Byte 0 (Command)

Byte 0 (Command)

Note: TEXT_NORTH is the default text orientation



See Also: PRINT_CHAR, PRINT_STRING, SELECT_FONT

Example:

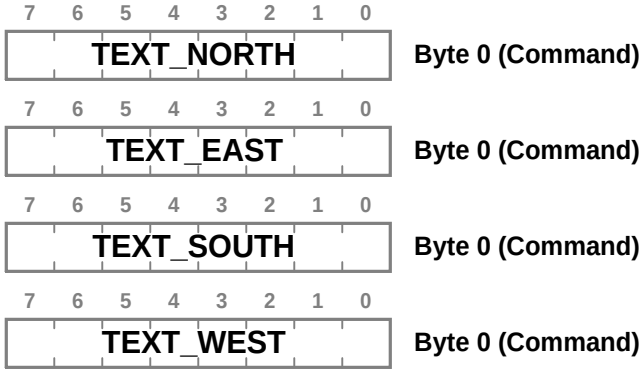
The following sequence will print the text pattern similar to the one on the picture above.

```
SET_XY         25 hex
60             60 dec
10             10 dec
SELECT_FONT    2B hex
0               0 dec
TEXT_NORTH     60 hex
PRINT_STRING   2D hex
"Text North "
```

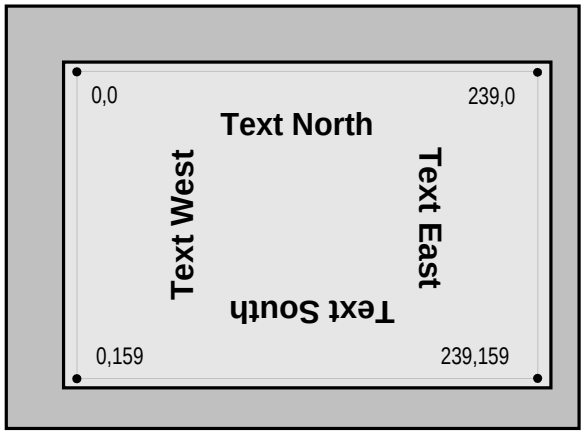
```
NULL            0 hex
TEXT_EAST      61 hex
PRINT_STRING   2D hex
"  Text East  "
NULL            0 hex
TEXT_SOUTH     62 hex
PRINT_STRING   2D hex
" Text South "
NULL            0 hex
TEXT_WEST      63 hex
PRINT_STRING   2D hex
"  Text West "
NULL            0 hex
```

2.5.27 TEXT_SOUTH

Description: Set the orientation of the text, as shown on the picture below
Class: Single Byte Commands
Code: **TEXT_NORTH**: **60**hex, **96**dec, **'** ASCII
TEXT_EAST : **61**hex, **97**dec, **a** ASCII
TEXT_SOUTH: **62**hex, **98**dec, **b** ASCII
TEXT_WEST : **2F**hex, **99**dec, **c** ASCII


7 6 5 4 3 2 1 0
TEXT_NORTH
Byte 0 (Command)
7 6 5 4 3 2 1 0
TEXT_EAST
Byte 0 (Command)
7 6 5 4 3 2 1 0
TEXT_SOUTH
Byte 0 (Command)
7 6 5 4 3 2 1 0
TEXT_WEST
Byte 0 (Command)

Note: TEXT_NORTH is the default text orientation



See Also: PRINT_CHAR, PRINT_STRING, SELECT_FONT

Example:

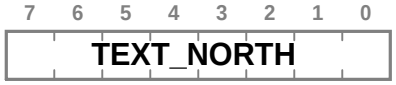
The following sequence will print the text pattern similar to the one on the picture above.

```
SET_XY          25 hex
60              60 dec
10              10 dec
SELECT_FONT     2B hex
0                0 dec
TEXT_NORTH      60 hex
PRINT_STRING    2D hex
"Text North "
```

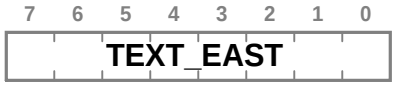
```
NULL            0 hex
TEXT_EAST      61 hex
PRINT_STRING   2D hex
"  Text East  "
NULL            0 hex
TEXT_SOUTH     62 hex
PRINT_STRING   2D hex
" Text South "
NULL            0 hex
TEXT_WEST      63 hex
PRINT_STRING   2D hex
"  Text West "
NULL            0 hex
```

2.5.28 TEXT_WEST

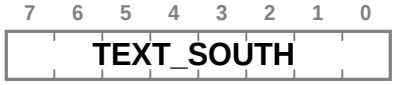
Description: Set the orientation of the text, as shown on the picture below
Class: Single Byte Commands
Code: **TEXT_NORTH**: **60**hex, **96**dec, **'** ASCII
TEXT_EAST : **61**hex, **97**dec, **a** ASCII
TEXT_SOUTH: **62**hex, **98**dec, **b** ASCII
TEXT_WEST : **2F**hex, **99**dec, **c** ASCII



Byte 0 (Command)



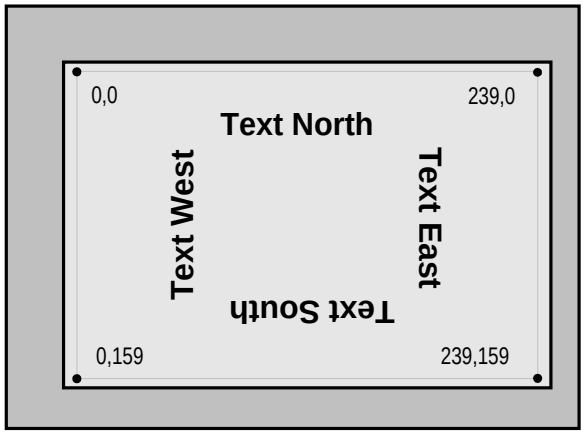
Byte 0 (Command)



Byte 0 (Command)

Byte 0 (Command)

Note: TEXT_NORTH is the default text orientation



See Also: PRINT_CHAR, PRINT_STRING, SELECT_FONT

Example:

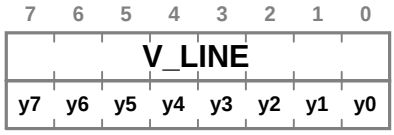
The following sequence will print the text pattern similar to the one on the picture above.

```
SET_XY          25 hex
60              60 dec
10              10 dec
SELECT_FONT     2B hex
0                0 dec
TEXT_NORTH      60 hex
PRINT_STRING    2D hex
"Text North "
```

```
NULL            0 hex
TEXT_EAST      61 hex
PRINT_STRING   2D hex
"  Text East  "
NULL            0 hex
TEXT_SOUTH     62 hex
PRINT_STRING   2D hex
" Text South "
NULL            0 hex
TEXT_WEST      63 hex
PRINT_STRING   2D hex
"  Text West "
NULL            0 hex
```

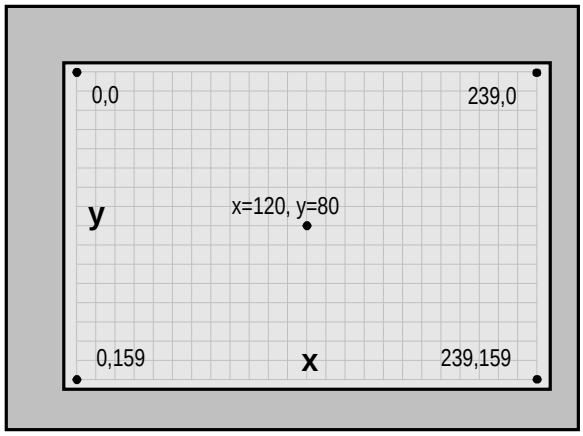
2.5.29 V_LINE

Description: Fast draws a vertical line from Current Position, to the row specified by the parameter.
Class: Double Byte Command
Code: **41**hex, **65**dec, **A** ASCII



Byte 0 (Command)
Byte 1 (Y)

Note: The screen size is 240x160. However, the valid Y range is 0 - 255



See Also: H_LINE, SET_XY

Example:

The following sequence will draw the vertical blue line from (95, 10) to (95, 110)

```
SET_COLOR    24 hex
BLUE       11000000 bin
SET_XY       25 hex
 95          95 dec
 10          10 dec
V_LINE       41 hex
110         110 dec
```

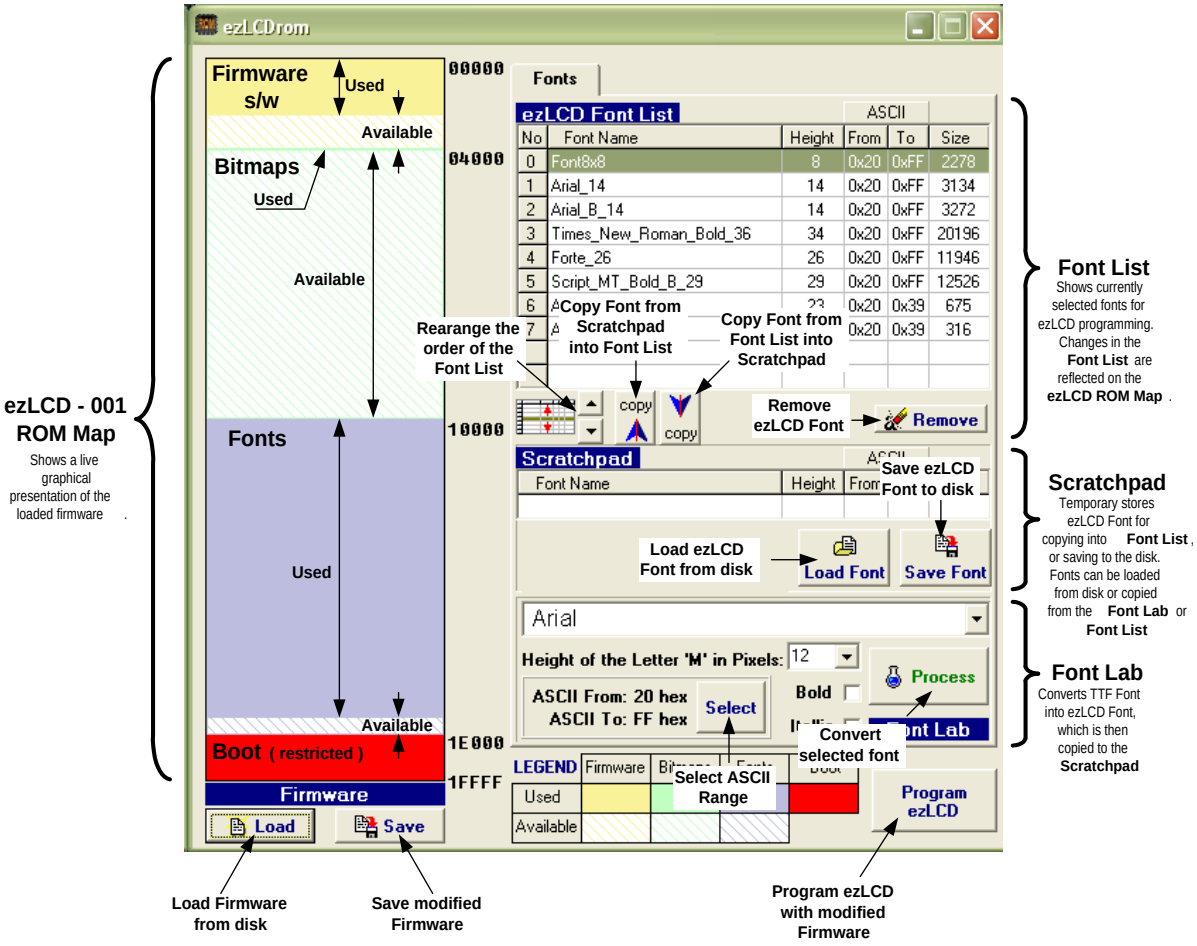
2.6 ezLCDrom Utility

2.6.1 Overview

The ezLCDrom is a utility, which allows the user to customize the Firmware of the ezLCD-002 by:

1. Adding and removing fonts
2. Adding and removing bitmaps or icons
3. Changing ezLCD settings like serial baudrate, pin assignments, etc.

Note: In this preliminary version only 1. is implemented

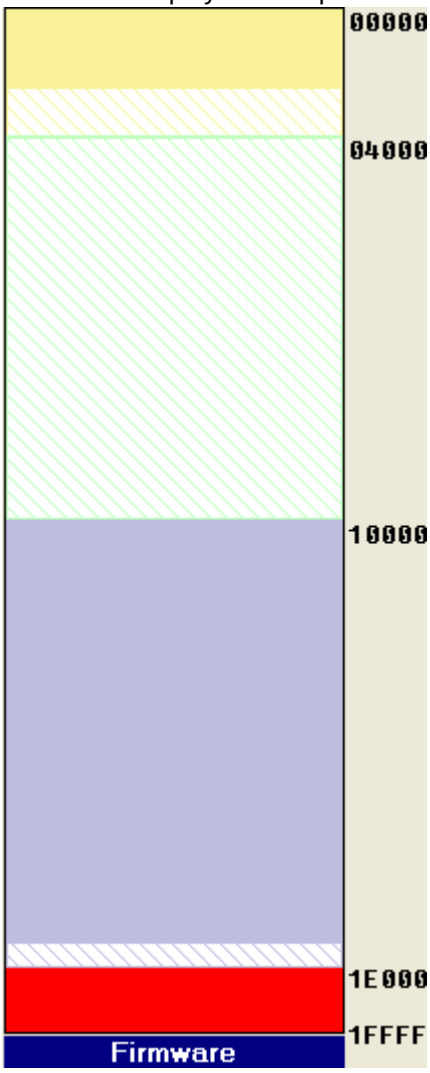


2.6.2 Loading Firmware file from disk

The ezLCD Firmware file is written in Intel Hex format and has an extension: `.hex`
To load the Firmware into ezLCDrom:

1. Click on Firmware **Load**
2. Select Firmware file

Upon loading the Firmware from disk,
ezLCDrom displays the Map of the ezLCD ROM:



Where:

00000 - 03FFF (16kB)
Space used by system and software.
Available space left for the system updates

04000 - 0FFFF (48kB)
Space used by bitmaps
Available space left for new bitmaps

10000 - 1DFFF (56kB)
Space used by fonts
Available space left for new fonts

1E000 - 1FFFF (8kB)
Boot space

2.6.3 Saving Firmware file

The ezLCD Firmware file will be written in Intel Hex format and should have an extension: `.hex`
To save the modified Firmware on disk:

1. Click on Firmware **Save**
2. Enter the filename and then press Save in the file save dialog

2.6.4 Programming ezLCD

To program the ezLCD with the modified Firmware:

Press **Program ezLCD**

This will:

- open a console window
- load a new firmware into the ezLCD

Example of messages displayed by the console during the successful programming:

```
Detecting.. AVRISP found on COM1:
Reading FLASH input file.. OK
Setting device parameters, serial programming mode ..OK
Entering programming mode.. OK
Erasing device.. OK
Programming FLASH using block mode..  100% OK
Leaving programming mode.. OK
```

2.6.5 How To

2.6.5.1 Add a new font to the ezLCD

To create and add a new font to the ezLCD:

1. Load the ezLCD Firmware from the disk, by pressing the

button.
2. Specify font parameters in the **Font Lab**
3. Select the ASCII Range of the font by pressing

button.
4. Press

to convert the selected TTF font into ezLCD font. Upon successful conversion, the new font will be displayed on the **Scratchpad**.
5. You can save the font by pressing

on the **Scratchpad**.
6. Rearrange the **ezlcd Font List**, if necessary.
7. Press

to add the **Scratchpad** font to the **ezLCD Font List**.
8. Rearrange the **ezlcd Font List**, if necessary.
9. You can save the new ezLCD Firmware by pressing the

button.
10. Program the ezLCD with the new Firmware.

2.6.5.2 Rearrange the fonts

To rearrange fonts on the ezLCD Font List:

1. Make sure that the ezLCD Firmware is loaded.
2. You can:

- Rearrange the order of fonts by pressing one of buttons.
- Remove the font from the list by pressing button.

2.6.5.3 Save a font from the ezLCD Font List

To save a font from the ezLCD Font List:

1. Make sure that the ezLCD Firmware is loaded.
2. Select the font for saving from the **ezLCD Font List**.
3. Press

to copy a font from the **ezLCD Font List** into the **Scratchpad**.
Caution: This will replace the current Scratchpad font.
4. Save the font by pressing

on the **Scratchpad**.

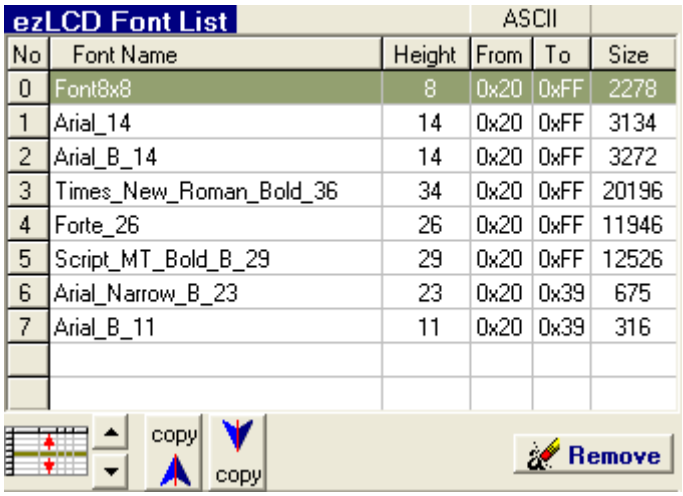
2.6.6 Fonts

2.6.6.1 ezLCD Font List

The **ezLCD Font List** is used to perform the following operations:

- Adding new fonts to the Firmware.
- Removing fonts from the Firmware.
- Rearranging the order of the Firmware fonts.

The **ezLCD Font List** shows the fonts of the loaded from the disk Firmware:



Where:

No - Font Number (used in the command SELECT_FONT)

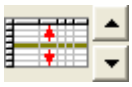
Font Name - Name of the Font (this is obvious)

Height - Distance (in ezLCD pixels) from the lowest point to the highest point of the font.

For example:

ASCII From and ASCII To - Limits of the ASCII Range. Letters outside the ASCII Range will not be drawn by the ezLCD. Minimizing the ASCII Range saves ezLCD ROM space.

Size - Number of bytes occupied by font

 - Rearrange the order of the fonts, by moving the selected font up or down

- Add the Scratchpad font to the end of the list.

- Copy the selected font to the Scratchpad, where it can be saved to the disk.

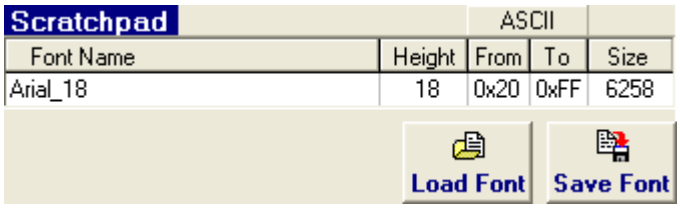
- Remove (delete, erase) the selected font from the list

2.6.6.2 Scratchpad

Scratchpad is used as an interfacing buffer between the disk, the **ezLCD Font List**, and the **Font Lab**.

Scratchpad Output:
- Adding the **Scratchpad** font to the **ezLCD Font List**
- Saving the **Scratchpad** font on the disk

Scratchpad Input:
- **Font Lab** puts newly generated font on the **Scratchpad**
- Adding the **Scratchpad** font to the **Font List**
- Loading an ezLCD font from the disk



Where:

Font Name - Name of the Scratchpad font (this is obvious)

Height - Distance (in ezLCD pixels) from the lowest point to the highest point of the font.
For example:

ASCII From and ASCII To - Limits of the ASCII Range. Letters outside the ASCII Range will not be drawn by the ezLCD. Minimizing the ASCII Range saves ezLCD ROM space.

Size - Number of bytes occupied by font

- Load a font from the disk

- Save the Scratchpad font on the disk

ezLCD Font List Scratchpad Operations:

- Add the Scratchpad font to the end of the ezLCD Font List

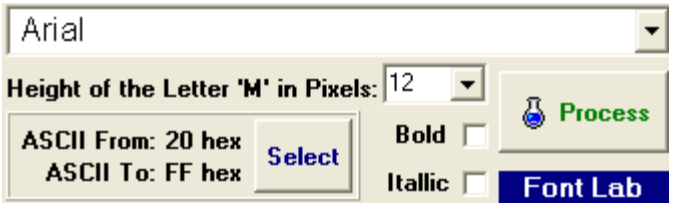
- Copy the selected font to the Scratchpad, where it can be saved to the disk.

Font Lab Scratchpad Operations:

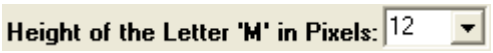
- Generate a new font and put it on the Scratchpad

2.6.6.3 Font Lab

Font Lab is used to convert TTF fonts into ezLCD fonts.
The converted font is moved to the **Scratchpad**.



Where:



Letter 'M' is used as a common reference to specify the font height.
Usually the font height will be bigger then letter M, since it is defined as the distance (in ezLCD pixels) from the lowest point to the highest point of the font, as it is shown on the example below.

However, for example, if the particular font contains only capital letters (ASCII Range: 41 to 5A hex), it's height will be equal to the height of the letter 'M'.

This panel is used to specify the ASCII range of the font.
Letters outside the ASCII Range will not be drawn by the ezLCD. Minimizing the ASCII Range saves ezLCD ROM space.

ASCII From: - Displays the bottom of the ASCII Range

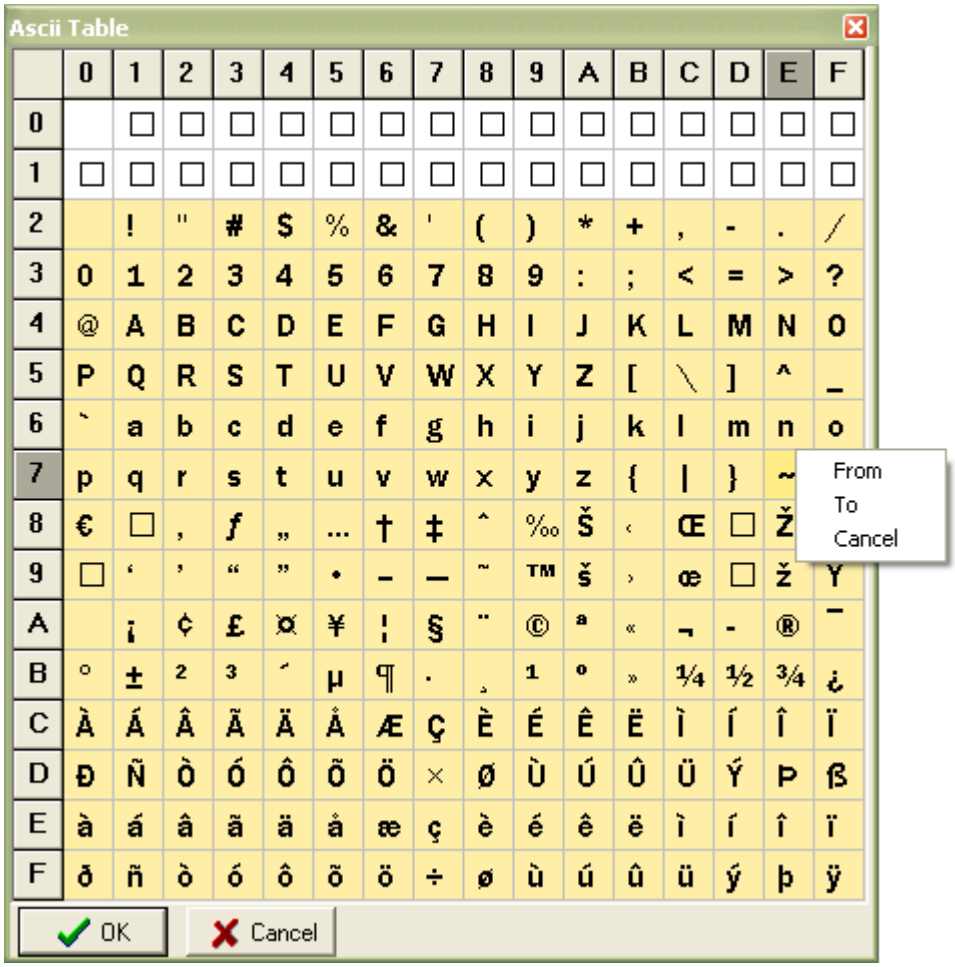
ASCII To: - Displays the top of the ASCII Range

- Selects the ASCII Range.
Described in Selecting ASCII Range

This button is used to start converting a TTF font into the ezLCD Font. The converted font is moved to the **Scratchpad**.

2.6.6.3.1 Selecting ASCII Range

When the **Select** button is pressed, the following form pop-ups:



The above form displays the ASCII Table of the selected font.

The currently selected ASCII Range has a background color:
The limits of the ASCII Range may be modified by clicking on the table cell.
If there is a need to distinguish between "From" and "To", ezLCDrom will display the following pop-up menu:

Press

to confirm the new ASCII Range,

or press

to return without any modifications.

Index

- 8 -

8 bit parallel 2

- A -

ARC 21
ASCII 66
ASCII From 65
ASCII Range 66
ASCII To 65
ATmega128 2

- B -

Baudrate 11
Block Diagram 4
BOX 23
BOX_FILL 24

- C -

CIRCLE_R 25
CIRCLE_R_FILL 26
CLS 27
CN1 11
Command Buffer 3
Command Interpreter 3
commands 2
Communication Parameters 10

- D -

Display 8
Drivers 11

- E -

EEPROM 11
ezLCD Font List 63
ezLCD-001 2
ezLCDrom 56

- F -

Font 63
Font Lab 65
fonts 63
FT232BM 11

- H -

H_LINE 28
Handshake 11

- I -

I2C 2

- L -

LCD 8
LCD controller 2, 3
Light 8
LIGHT_OFF 29
LIGHT_ON 30
LINE_TO_XY 31
Linux 11

- M -

MPU 10

- N -

No of Stop Bits 11

- O -

OSX 11

- P -

Parity 11
PICTURE 32
PLOT 33
PLOT_XY 34

Power 8
PRINT_CHAR 35
PRINT_CHAR_BG 36
PRINT_STRING 37
PRINT_STRING_BG 38
PUT_BITMAP 39

- R -

ROM 57
RS232 2

- S -

Scratchpad 64
SED1375 2
SELECT_FONT 43
SET_BG_COLOR 44
SET_COLOR 45
SET_XY 46
SPI 2

- T -

TEXT 47
TEXT_EAST 47
TEXT_NORTH 47
TEXT_SOUTH 47
TEXT_WEST 47

- U -

USB 2

- V -

V_LINE 55
Video RAM 4

- W -

Windows 11